REMARKS

Applicant is in receipt of the Office Action mailed April 19, 2007. Claims 1-42 have been cancelled. New claims 43-68 have been added. Thus, claims 43-68 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Amendments

Applicant has cancelled all the previous claims, and has submitted new claims that Applicant feels more clearly represent Applicant's invention.

Section 102 Rejections

Claims 1, 33-36, and 38-40 were rejected under 35 U.S.C. 102(b) as being anticipated by Kudokoli et al. (US Pat. Pub. 2001/0024211, "Kudokoli"). Applicant has cancelled these claims, thus rendering their rejections moot. Applicant believes that the new claims are patentably distinct and non-obvious over Kudokoli.

As the Examiner is certainly aware, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

New claim 43 recites:

43. A computer-accessible memory medium that stores program instructions executable by a processor to perform:

displaying a data acquisition (DAQ) node in a graphical program;

receiving first user input invoking display of a plurality of DAQ functions for the DAQ node;

displaying the plurality of functions for the DAQ node in response to the first user input;

receiving second user input selecting a function from the plurality of DAQ functions;

determining graphical program code based on the second user input, wherein the determined graphical program code is executable to provide functionality in accordance with the selected function;

associating the determined graphical program code with the DAQ node, wherein, when the DAQ node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.

Nowhere does the cited art disclose **determining graphical program code based on the second user input, wherein the determined graphical program code is executable to provide functionality in accordance with the selected function; and associating the determined graphical program code with the DAQ node, wherein, when the DAQ node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**.

It should be noted that in the present application, "associating the determined graphical program code with the DAQ node" refers to configuring the node to include the determined program code (see, e.g., p.8, lines 16-20). In other words, the node is configured with graphical program code that was not previously included in the node. None of the cited references disclose this feature.

Note that, as described in Kudokoli's Abstract, Kudokoli is directed to programmatically generating and modifying graphical programs in response to receiving program information. Specifically, Kudokoli describes various components, e.g., nodes, of a graphical program generation (GPG) program that, when executed, generates another graphical program, where nodes in the GPG program each execute to create or manipulate objects in the new graphical program.

Applicant respectfully submits that Kudokoli, e.g., cited Figure 25A, section 2, [0278], Figure 25B, section 5, [0281], and Figures 25A-D, sections 1-13 (and Kudokoli in general), does not disclose or suggest such determination of graphical program code in response to user input selecting a DAQ function for the node, and associating the graphical program code with the DAQ node so that when the DAQ node executes, the graphical program code is executed.

For example, note that Kudokoli's New VI Object Reference node executes to create new objects, e.g., user interface controls, graphical program nodes, etc., but these new created objects are part of a newly created graphical program, and are not represented by the New VI Object Reference node, nor do they comprise graphical program code that is then associated with the New VI Object Reference node and executed with the New VI Object Reference node executes in the graphical program. Similar arguments apply to Kudokoli's other described nodes.

The Office Action also cites Kudokoli [0213], [0215], [0217], and Figure 22, particularly with regard to Kudokoli's New VI Object Reference node. However, Applicant has reviewed the cited portions carefully, and respectfully submits that in Kudokoli's description of this node (and others), no mention is made of the user selecting a function for the DAQ node, and graphical program code being determined (automatically) that implements the selected function, nor associating the determined graphical program code with the node.

Rather, in Kudokoli's system, the user input specifies a *configuration* of the node's inherent functionality, e.g., by selecting configuration options that specify input parameters to the node/function. In other words, the graphical program code are already associated with or belong to the node, and the user simply configures it to create the desired kind of object.

Figures 21 and 22 makes this clear, in that the displayed menus are configuration menus that configure the functionality already inherent in the node. Associated text from [0215] states" "FIG. 21 illustrates how a user may choose a value for the vi object class input by selecting from a hierarchical menu." Similarly, [0217] states: "FIG. 22 illustrates how a user may choose a value for the style input by selecting from a hierarchical menu". Nowhere does Kudokoli teach or suggest selection or specification

of a function (or function type), determination of graphical program code implementing the selected function, nor associating such determined graphical program code with the node, where, when the node executes in the graphical program, the determined graphical program code are operable to execute to provide the functionality in accordance with the selected function.

Applicant further nodes that Kudokoli's nodes are not DAQ nodes, nor are the functions discussed in Kudokoli DAQ functions.

Thus, for at least the above reasons, Applicant respectfully submits that Kudokoli fails to disclose these features of claim 43.

Applicant further submits that Kudokoli also fails to disclose **displaying the plurality of functions for the DAQ node in response to the first user input.** The Office Action cites [0215]- [0216] and Figure 21 in asserting that Kudokoli discloses this feature. However, Applicant respectfully submits that the cited text and figures fail to disclose this feature.

Applicant respectfully submits that the menus of Figure 21 present configuration options for the node functionality, i.e., the displayed menus of Figure 21 are configuration menus that configure the functionality *already inherent* in the node. Nowhere does Kudokoli disclose displaying *functions* for the node. This point is made clearly in [0215] which states: "FIG. 21 illustrates how a user may choose a value for the vi object class input by selecting from a hierarchical menu."

Thus, Kudokoli fails to teach or suggest this feature of claim 43.

Applicant further submits that Kudokoli also fails to disclose **receiving second user input selecting a function from the plurality of DAQ functions**, as recited in claim 43.

Cited paragraph [0217] and Figure 22 (and Kudokoli in general) in no way disclose this feature. For example, these citations are directed to inputs and outputs of the New VI Object Reference node, discussed above, and nowhere do they mention or even hint at user selection of a DAQ function as claimed.

Thus, Kudokoli fails to teach or suggest this feature of claim 43.

Thus, for at least the reasons provided above, Applicant submits that claim 43 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 52 includes similar limitations as claim 43, and so the above arguments apply with equal force to this claim. Thus, claim 52 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 53 and 58 each includes similar limitations as claim 43, but where a second node is determined based on the selected function, where the second node includes graphical program code executable to provide functionality in accordance with the selected function, and where the DAQ node in the graphical program is replaced with the second node, where, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function. In other words, rather than associating the determined graphical program code with the original DAQ node, a node that includes the determined graphical program code is used in lieu of the original DAQ node. Nowhere does Kudokoli teach or suggest these features.

Thus, for at least the reasons provided above, Applicant submits that claims 53 and 58, and those claims respective dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 59 and 66 each includes similar limitations as claim 43, but where, instead of user input invoking display of the DAQ functions, and user input selecting the function, user input specifies one or more inputs to the DAQ node, e.g., wires and configures inputs to the DAQ node, in response to which the function is automatically selected, and the graphical program code determined and associated, as with claim 43. In other words, the graphical program code is automatically determined and associated with the DAQ node based on inputs to the node. Nowhere does Kudokoli disclose these features.

Thus, for at least the reasons provided above, Applicant submits that claims 59 and 66, and those claims respective dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 67 and 68 each includes similar limitations as claim 59, where user input specifies one or more inputs to the DAQ node, e.g., wires and configures inputs to the DAQ node, in response to which the function is automatically selected. However, in claims 67 and 68, a second node is determined based on the selected function, where the second node includes graphical program code executable to provide functionality in accordance with the selected function, and where the DAQ node in the graphical program is replaced with the second node, where, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function, as in claim 53.

Nowhere does Kudokoli disclose these features.

Thus, for at least the reasons provided above, Applicant submits that claims 67 and 68, and those claims respective dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Thus, Applicant respectfully submits that claims 43-68 are patentably distinct and non-obvious over Kudokoli and are thus allowable.

Claim 37 was rejected under 35 U.S.C. 102(e) as being anticipated by Parthasarathy et al (US Patent No. 6,064,812, "Parthasarathy"). Applicant has cancelled claim 37, rendering its rejection moot.

However, Applicant respectfully submits that Parthasarathy does not teach or suggest the limitations of Applicant's new claims.

For example, nowhere does Parthasarathy disclose **displaying the plurality of DAQ functions for the DAQ node in response to the first user input**, as recited in claim 37.

The Office Action cited Figure 12, Figure 6b, and col.16:29-col.17:43, particularly, the "Application/Property node", asserting that the "Application/Property

node" replaces an automation Property node. First, Applicant respectfully submits that the Application/Property node, actually called an Excel Application automation property node in the cited text, *is* an automation property node, according to Parthasarathy, and is nowhere described as replacing an automation property node. Moreover, as Parthasarathy makes clear, the automation property node displays *properties* of a specified automation class, and is nowhere described as displaying DAQ functions for a node. Moreover, Parthasarathy's Application/Property node is not a DAQ node.

Thus, Parthasarathy fails to teach or suggest this claimed feature.

Nowhere does Parthasarathy disclose **receiving second user input selecting a function from the plurality of DAQ functions**, nor **determining a second node based on the selected function, wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function**, nor **replacing the DAQ node in the graphical program with the second node, wherein, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**, as claimed.

The Office Action cites Figure 17, Application/Workbook, col.20:23-40, and Figure 6c:98b and 122, and col.18:53-67. Applicant has reviewed these citations carefully, and respectfully submits that col.20:23-40 is directed to invoking help information regarding a method of an application, and nowhere discloses these features. Figure 6c:98b and 122, and col.18:53-67 are directed to an automation invoke node that operates to invoke a user-selected method of an automation class. More specifically, per Parthasarathy, the invoke node is configured to invoke the specified method for the automation class.

Nowhere do the citations disclose determining a second node based on the selected function, where the second node comprises graphical program code executable to provide functionality in accordance with the selected function, nor in the graphical program with the second node, where, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.

As has been explained previously, the construction of execution instructions for the graphical program described by Parthasarathy is actually a compilation step for the graphical program, as described in col.19:38-47:

> Once the user has created the VI 50, the user instructs the graphical programming environment to construct execution instructions in accordance with the nodes in the block diagram in step 126. Preferably, constructing the execution instructions comprises generating machine language instructions into an executable program. Alternatively, constructing the execution instructions comprises generating programming language instructions, such as C language instructions, and compiling the programming language instructions into an executable program.

Thus, Parthasarathy does not disclose determining a second node nor determining graphical program code implementing the selected function.

Nor does Parthasarathy disclose **associating the determined graphical program code with the DAQ node, wherein, when the DAQ node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.**

As noted above, in the present application, "associating the determined graphical program code with the DAQ node" refers to configuring the node to include the determined program code (see, e.g., p.8, lines 16-20). In other words, the node is configured with graphical program code that was not previously included in the node. Parthasarathy nowhere discloses this feature. Applicant notes that Parthasarathy's invoke node obtains a reference to a specified object (an instantiated automation node) and invokes methods of that object via the reference. This is quite different from including the code in the node itself, i.e., making that code the node's code, since in Parthasarathy, the object's method is called by proxy, but the method does not properly belong to the invoke node.

Thus, Parthasarathy fails to disclose these claimed features.

Thus, for at least the reasons provided above, Applicant submits that Parthasarathy fails to teach or suggest all the features and limitations of Applicant's new claims, and so these new claims and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Section 103 Rejections

Claims 1, 3-30, 32-36, and 38-42 were rejected under 35 U.S.C. 103(a) as being unpatentable over to Parthasarathy et al. (US Patent No. 6,064,812, "Parthasarathy") in view of Dye et al. (US Patent No. 6,102,965, "Dye"). Applicant has cancelled these claims, thus rendering their rejections moot. Applicant believes that the new claims are patentably distinct and non-obvious over Parthasarathy and Dye.

As discussed above, Parthasarathy fails to disclose all the features and limitations of Applicant's new independent claims. Regarding Parthasarathy's automation invoke node, as noted above, in the present application, "associating the determined graphical program code with the DAQ node" refers to configuring the node to include the determined program code (see, e.g., p.8, lines 16-20). In other words, the node is configured with graphical program code that was not previously included in the node. Parthasarathy's invoke node obtains a reference to a specified object (an instantiated automation node) and invokes methods of that object via the reference. This is quite different from including the code in the node itself, i.e., making that code the node's code, since in Parthasarathy, the object's method is called by proxy, but the method does not properly belong to the invoke node.

Moreover, Dye fails to provide the various claimed features and limitations of the independent claims not taught by Parthasarathy.

Thus, Applicant respectfully submits that claims 43-68 are patentably distinct and non-obvious over Parthasarathy and Dye, taken singly or in combination, and are thus allowable.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-80201/JCH.

Also filed herewith are the following items:

☒ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Mark S. Williams/

Mark S. Williams, Reg. #50,658
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: July 19, 2007 JCH/MSW